

Bolt Beranek and Newman Inc.



12

Report No. 4396

LEVEL

1086123

ADA 086132

Development of a Voice Funnel System

Quarterly Technical Report No. 5
1 August 1979 to 31 October 1979



June 1980

Prepared for:
Defense Advanced Research Projects Agency

bug FILE COPY,

DISTRIBUTION STATEMENT A
Approved for public release
Distribution Unlimited

80 6 30 140

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A086132	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) DEVELOPMENT OF A VOICE FUNNEL SYSTEM. QUARTERLY TECHNICAL REPORT NO. 5 1 Aug-31 Oct 79		5. TYPE OF REPORT & PERIOD COVERED Quarterly Technical
7. AUTHOR(s) M./Hoffman		6. PERFORMING ORG. REPORT NUMBER 43961
8. PERFORMING ORGANIZATION NAME AND ADDRESS Bolt Beranek and Newman Inc. 50 Moulton Street, Cambridge, MA 02138		9. CONTRACT OR GRANT NUMBER(s) MDA903-78-C-0356 ARPA Order-3653
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Blvd., Arlington, VA 22209		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS ARPA Order No. 3653
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE June 1980
		13. NUMBER OF PAGES 18
16. DISTRIBUTION STATEMENT (of this Report) DISTRIBUTION UNLIMITED		15. SECURITY CLASS. (of this report) 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Voice Funnel, Digitized Speech, Packet Switching, Butterfly Switch, Multiprocessor		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This Quarterly Technical Report covers work performed during the period noted on the development of a high-speed interface, called a Voice Funnel, between digitized speech streams and a packet-switching communications network.		

DD FORM 1 JAN 79 1473

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

060100

bpg

Report No. 4396

Bolt Beranek and Newman Inc.

DEVELOPMENT OF A VOICE FUNNEL SYSTEM

QUARTERLY TECHNICAL REPORT NO. 5
1 August 1979 to 31 October 1979

June 1980

This research was sponsored by the
Defense Advanced Research Projects
Agency under ARPA Order No.: 3653
Contract No.: MDA903-78-C-0356
Monitored by DARPA/IPTO
Effective date of contract: 1 September 1978
Contract expiration date: 30 November 1980
Principal investigator: R. D. Rettberg

Prepared for:

Dr. Robert E. Kahn, Director
Defense Advanced Research Projects Agency
Information Processing Techniques Office
1400 Wilson Boulevard
Arlington, VA 22209

The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either express or implied, of the Defense Advanced Research Projects Agency or the United States Government.

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DDC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/_____	
Availability Codes	
Dist.	Avail and/or special
A	

QUARTERLY TECHNICAL REPORT NO. 5

Contents

1. Introduction	1
2. Object Management System	2
2.1 Use of the Object Management System	3
2.2 Structure and System Integration	5
3. System Power Supply	11

1. Introduction

This Quarterly Technical Report, Number 5, describes aspects of our work performed under Contract No. MDA903-78-C-0356 during the period from 1 August 1979 to 31 October 1979. This is the fifth in a series of Quarterly Technical Reports on the design of a packet speech concentrator, the Voice Funnel.

This report focuses on two aspects of the system: the development of the Object Management System, a part of our software effort, and the development of a suitable power supply for a reliable hardware system. In the following sections we present some of the results of this work.

2. Object Management System

The purpose of the Object Management System is to manage access to system resources (such as memory segments, I/O buffers, and processes). It provides an environment in which objects are given types, names, and protection to aid the programmer, and makes the virtual-to-physical memory mapping largely invisible.

The Object Management System brings together a variety of services which most operating systems provide in a less cohesive manner. Many operating systems include a file system for secondary storage organization and access. The file system often includes facilities for naming and sharing files, and for keeping type information about them so that access can be provided in a controlled way. Similar services for objects in primary memory are usually less elaborate, but usually include memory protection and any necessary memory mapping.

The Voice Funnel does not need secondary storage and therefore will not have the capabilities of a file system. The Object Management System provides some of the file system features for "objects" which reside in primary memory. In addition, memory allocation, memory protection, and controlled sharing are provided through the Object Management System.

2.1 Use of the Object Management System

The user views the system as a collection of resources, called objects, which the Object Management System manipulates on his behalf and under his control. Examples of important object types are the user's process and memory segments. The user identifies objects by OIDs (Object IDentifiers), which are location-independent tags assigned by the Object Management System and through which the Object Management System can locate the object. For convenience, a user may associate a character string as the name for the object, just as he would name a file in a conventional file system. An object can be located either from its OID or, more slowly, from its name.

An OID is a 32-bit quantity, consisting of three fields. Two of the fields, an 8-bit processor number and a 16-bit index, are used to locate the Object Attribute Block (OAB) for the object. Each processor node contains an OAB table, and the index selects an OAB from this table. The third field is an 8-bit check field which is matched against the corresponding field in the OAB at the processor node and offset selected by the OID. Over time an OAB may be used for a succession of distinct objects. Each time an object is destroyed, its OAB check field is incremented, so that a new object using the OAB will have a distinct value in the check field. Since there are 256 possible values for the check field, the probability of an OID for a

deleted object matching the check field is quite small. We expect this to be a powerful debugging facility.

The OAB is the minimum system structure associated with an object. It consists of a collection of fields describing the object. Included in the OAB are the check field, type of object field, and pointers to the object's representation in memory and to the long name of the object. Note that either of these pointers may be null. Other fields of the OAB are used for object status, protection, and debugging information, as well as for highly protected type specific information.

A user wishing to use an object must first obtain permission from the Object Management System. If, for instance, he wishes to write an I/O buffer, he must pass its OID to a routine which places it in his address space; the user either specifies the virtual address he wishes it to have in his environment, or if he fails to specify the address, receives the virtual address back from the Object Management System.

The user must also specify the access mode (e.g., user writable) required for his usage. This access mode must be a subset of the access which he is permitted for the object, or his request will be rejected. If the user attempts to access it in a mode other than what he has requested, the hardware of the Memory Management System blocks the access. The user may request those access rights which he expects to use, rather than the full

access to which he may be entitled. By this means unintended object references will often be recognized by the Object Management System as errors, making it easier to find software design and coding errors.

When the user is through using the object, he can ask the Object Management System to remove the object from his address space, and then may no longer reference the object. Of course, this does not prejudice his future use of the object in the same or a different mode. In this way, the user may alter the access permission to an object over time in order to match the protection to his requirements on a moment-by-moment basis.

The above description is oriented to objects which are regions of memory (logical segments or buffers), for which the hardware is capable of providing the appropriate protection. More complex objects, such as processes, may not be directly readable or writable by the user at all. For these objects, the object system will be augmented with routines which manipulate the objects on the user's behalf. The same protection mechanism will be used to control what operations these software routines will permit on the object.

2.2 Structure and System Integration

The services of the Object Management System enhance the structure and protection of the operating system. We would

therefore like to keep the Object Management System at a sufficiently low level to permit the remainder of the operating system to treat it as an available resource. However, the Object Management System also utilizes services from other parts of the operating system, and therefore we might want to construct it above the remainder of the operating system.

In order to provide both the operating system and Object Management System with the services of the other in a reasonably efficient manner, we are constructing the Object Management System in layers. Primitive functions, such as physical memory allocation, lie below much of the rest of the operating system, while higher level functions, such as virtual memory allocation, are found in a much higher layer, above much of the rest of the operating system. Thus the various layers of the operating system see different degrees of sophistication in the Object Management System. Indeed, the higher layers of the Object Management System will call the same lower level Object Management System routines as do other parts of the operating system. This provides operating system services to the Object Management System while it provides Object Management System services to the rest of the operating system.

The Object Management System has four layers. The lowest layer, the Physical Memory Manager, is a memory allocator based on the physical machine characteristics. The next layer, the

Central Object Utility, is the heart of the Object Management System. It provides type-independent services on objects, and insulates the layers above it from the locality properties of the physical memory system of the Butterfly Multiprocessor. Above the Central Object Utility lies the Memory Mapping Manager. It maintains the mapping registers on the Butterfly Multiprocessor to support the Virtual Address space. The final layer of the Object Management System provides the services particular to specific types of objects, and provides an interface to the lower level functions.

The Physical Memory Manager is the lowest layer of the Object Management System. It contains a physical memory allocator which allocates memory in discrete sized blocks which correspond to multiples of the page size of the Butterfly Multiprocessor, and which can be easily protected by the Memory Manager hardware. This allocator can only allocate memory in a specified processor node; that is, each request for memory allocation must specify a processor node from which the memory is to be allocated, and the allocation will fail if there is not enough memory in the processor node specified.

A second allocator is provided which obtains blocks of memory from the main allocator and subdivides the memory to provide memory in arbitrarily small increments. However, these smaller allocations cannot be individually protected by the Object Management System.

The second level of the Object Management System is the Central Object Utility. It consists of a large number of procedures which perform all the type-independent operations on objects, such as creation, destruction, naming, locating and accessing, and fetching and altering the parametric descriptors of objects. It calls on the Physical Memory Manager to acquire and release memory for both the objects themselves and for its own needs.

In order to insulate higher layers of software from locality considerations of the physical memory system, the Central Object Utility manipulates location-independent object identifiers and names, and operates on data structures across nodes of the Butterfly Multiprocessor. Since all programs above the layer of the Central Object Utility use these identifiers to reference objects, programs in higher layers are effectively insulated from location-dependent features of the hardware. However, the user may occasionally want to specify the location of an object, since the time required to access the object depends on its location. For this reason, the Central Object Utility permits the user to control the processor node on which an object resides.

The Memory Mapping Manager is the third layer of the Object Management System. It contains the routines which maintain the memory maps for processes. Each entry of the memory map contains the physical address and protection attributes for accessing a

single object from a given process. The Memory Mapping Manager thus defines the virtual address space for all of the system above it.

The top level of the Object Management System is the set of type-specific object managers, the most important of which is the Virtual Segment Manager. The Virtual Segment Manager provides the user with virtual memory segments, which he may identify by their segment numbers.

Other objects will be supported by specific type managers, including processes, buffers, and dual queues. The set of type managers is designed to be expandable, and more type managers will be added as required. Each object manager will support calls to create, destroy, and modify objects of the appropriate type. In addition, there must be support either for mapping the object into the user's virtual space or, alternatively, to read and write the object directly. For example, the buffer manager will allow the user to map in the data portion of the I/O buffers, but will employ service calls to modify the buffer control information, so that it may check the validity of the control information.

Creating and destroying objects are expected to be relatively expensive operations for objects whose creation implies allocation of memory. For these objects, it may well be appropriate to keep the objects on free lists, and recycle them.

For example, I/O buffers will almost certainly be maintained as objects between uses, although creating and destroying them as needed might be more straightforward. If necessary, support for these free lists could be built into the type managers, making the activity invisible to the user program.

3. System Power Supply

As the computer industry knows well, the energy supply of a system is often surprisingly complex and expensive. Trouble comes in almost every form. For example, the source of power is unreliable, noisy, mechanically awkward, and lethal. Refining the power so that it may be used with integrated circuits requires a discipline different from logic design: the components are prone to various forms of breakdown and fatigue, and the mechanical and magnetic considerations are prominent. Finally, the power dissipated in the logic circuits and the supply itself must be removed from the machine.

In the Butterfly Multiprocessor we plan to design and build our own supplies and mount them on the same cards as the logic. The power supply structure we have in mind is a simple switching supply based on available regulator ICs and a MOSFET power transistor. Some of the reasons we have come to this choice include the following:

- Reliability - If a single power supply supplies more than one system component, the system is vulnerable to failure in that component. If we try to duplex centralized power supplies, the simple diode-coupling method being used now will not work because of variations in the voltage drop across the diode. This leads to on-board regulators after the diode.
- Modularity - By distributing the power regulation throughout the system, and relaxing the requirements for raw power distribution, we have vastly improved the modularity of the system. Modularity is a problem because it is often necessary to couple the power supply to only a few active components. Unfortunately it is then necessary to configure

that supply for the worst case power load. In the current configuration, the supply always matches the load. The astute reader will have noticed that the source of raw power is still centralized and must be configured for worst case; however, the cost of raw power is much less than the cost of regulated power and the wide input supply tolerances have eased the problems of configuring the raw power supply.

- Efficiency - The on-board supplies can be very efficient because they are switching supplies and because they are designed for the application at hand.
- Accuracy - Power supplies are often furnished with remote sensing connections to permit the supply to compensate for the voltage drop in the distribution system. Since the regulator is being placed on the card, these effects, as well as the effect of connector imperfections, will be much less important. The regulator being designed should be very good at supplying high quality DC to the circuitry.
- Current - By placing the regulator on the card, the amount of current supplied by each power supply circuit is small, making the design easier and more certain. By distributing the power at a higher voltage, the current flow in the distribution circuit can be reduced, and the size of power storage capacitors in the raw power supply can be reduced.
- Multiple Voltages - Since the supply is distributed, any multiple voltages can be supplied as required.
- Battery Backup - The high input voltage tolerance of the on-card regulator will make battery switch-over very simple. On-card regulation also is well suited to supplying backup power only to those parts of the machine which require power in a "holdup" configuration. For example, we can provide battery power to only the memory cards, since they have their own supply.

The power system is illustrated in Figure 1. As can be seen, the power supply is distributed throughout the machine in the form of a regulator on each card supplied by diode-coupled raw voltage at about 28 VDC. Because of diode-coupling, the source of raw DC can be implemented in a redundant manner, and indeed, if a battery holdup were required, it could be coupled in as well.

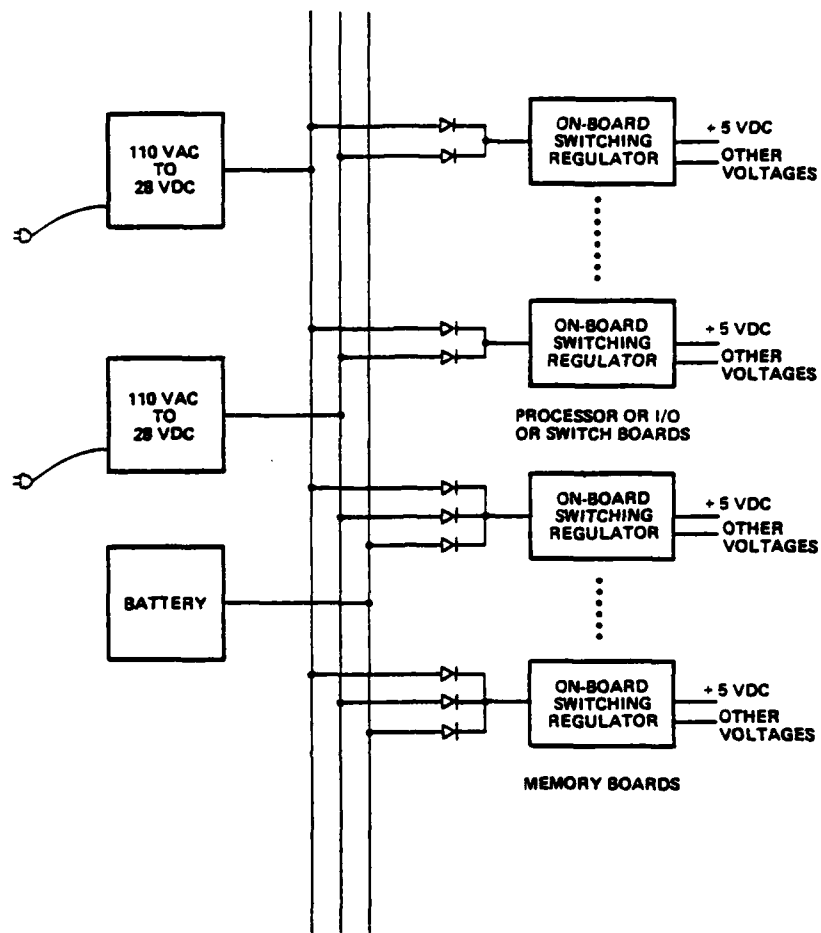


Figure 1 Power Distribution System

Supplying the raw power at 28 volts instead of 5 volts reduces the current flow in the power distribution system by a factor of 5 and reduces the losses in the diode coupling similarly. Since the on-board regulators are switching regulators, they can be designed to accept a wide variation in

input supply voltage. This is particularly important when a simple battery switch-over mechanism such as diode coupling is desired.

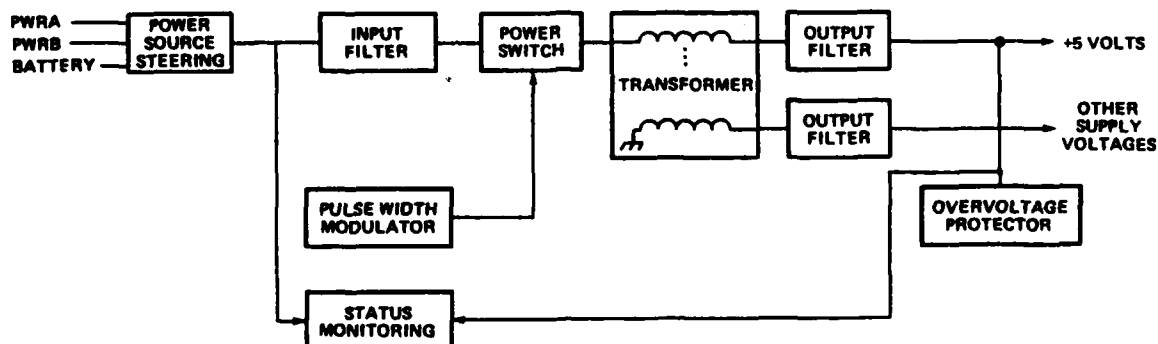


Figure 2 Block Diagram of On-Board Regulator

Each on-card regulator has the same set of functional blocks (as seen in Figure 2) although several of the blocks may be implemented with different components because of individual board voltage, current, and ripple requirements. The functional blocks are:

- Power Source Steering - This block consists of two or three diodes to allow diode ORing of the two separate power buses and, where battery holdup is required, the battery power bus. A fuse and inrush current limiter are also provided to protect the power sources from a faulty regulator.

- Input Filter - A T-section low pass filter reduces coupling of the high frequency (150 khz) current pulses into the power source.
- Power Switch - A power MOSFET switch is used to simplify the drive requirements and allow efficient high frequency operation.
- Transformer - The transformer can generate multiple output voltages of either polarity.
- Output Filter - The low pass PI-section output filter attenuates the ripple to acceptable limits.
- Pulse Width Modulator - An integrated circuit together with a few discrete components provides all the control circuitry for pulse width modulating the power switch to achieve regulation of the main output voltage. The chip contains voltage reference, error amplifier, oscillator, pulse width modulator, and shut-down circuitry.
- Status Monitoring - Comparators check that the power bus voltage is greater than the minimum voltage needed, that the output current is less than the overcurrent limit, and that the output voltages are within their correct tolerances.
- Overvoltage Protector - A commercial overvoltage protector on the main +5 supply assures that a catastrophic power supply failure will not damage the ICs.

Table 1 details the parts count, cost, board area percentage, output power, voltages required, and efficiency of the four on-board regulators thus far designed. The design effort for each power supply has averaged about 2 weeks including design, parts selection, prototype performance validation, and PC board layout and checking. The ripple and voltage regulation of the prototypes have basically conformed to the design equations. Thermal management has been very straightforward because of the high efficiency and modest power requirements of a single board.

	I/O Board	Switch Node Board	Memory Board	Processor Node Board
Number of non-mechanical parts	57	66	65	62
Parts Cost (\$)	52	59	59	54
Board Space Used (sq. in.)	30	24	22	24
Fraction of total Board Space	15%	19%	20%	11%
+5.0 volt current (amps)	6.5	1.8	1.0	9.3
+12.0 volt current (amps)	0.0	0.0	0.6	0.0
-2.7 volt current (amps)	0.0	1.2	0.0	0.0
-5.0 volt current (amps)	0.0	0.0	0.02	0.0
-5.2 volt current (amps)	0.2	0.6	0.0	0.5
Efficiency at 24 volts input	75%	72%	77%	76%
Battery backup	no	no	yes	no
Input power (watts)	45	18	16	65

Table 1 Summary of Regulator Characteristics

One issue of concern is power supply reliability. If more manpower and time were available, accelerated life testing could give information on any failure modes and component failure rates. Protection in the form of overvoltage and overcurrent detection with fast blow fuses will hopefully prevent regulator failures from propagating to other boards or destroying ICs.

Due to the high operating frequency and very fast switching transitions, Radio Frequency Interference could be a problem.

The layout of each regulator is such that a conductive cover can be placed over all the regulator components. It is hoped that the regulator cover in conjunction with the rack enclosure will attenuate the interference to an acceptable level.

DISTRIBUTION OF THIS REPORT

Defense Advanced Research Projects Agency

Dr. Robert E. Kahn (2)

Defense Supply Service -- Washington

Jane D. Hensley (1)

Defense Documentation Center (12)

Bolt Beranek and Newman Inc.

Library

Library, Canoga Park Office

R. Bressler

R. Brooks

P. Carvey

P. Castleman

F. Heart

M. Hoffman

M. Kraley

W. Mann

J. Pershing

R. Rettberg

E. Starr

E. Wolf